(3)

جدولة المعالج CPU Scheduling

مقدمة

جدولة المعالج هي إحدى الوظائف الأساسية في نظم التشغيل، حيث تهدف إلى تخصيص وقت المعالج (CPU) للعمليات (Processes) بطريقة فعالة وعادلة، مما يحقق أقصى استفادة من الموارد ويحسن أداء النظام في الاستجابة، الإنتاجية، والعدالة.

تتضمن جدولة المعالج اتخاذ قرارات حول أي عملية يتم تنفيذها، متى تُنفذ، وكم مدة التنفيذ.

في هذه المحاضرة، سنركز على المفاهيم الأساسية لجدولة المعالج، لفهم تطبيقاتها في نظم التشغيل:

1. انفجار المعالج (CPU Burst).
2. انفجار الإدخال/الإخراج (I/O Burst).
3. المجدول (Scheduler).
4. طوابير الجدولة (Scheduling Queues).
5. الجدولة الاستباقية وغير الاستباقية (Preemptive vs. Non-Preemptive Scheduling).

1. انفجار المعالج CPU Burst

هو الفترة الزمنية التي تحتاجها العملية لاستخدام المعالج بشكل مستمر لتنفيذ التعليمات دون الحاجة إلى عمليات إدخال/إخراج (I/O)، خلال هذه الفترة، تكون العملية في حالة التنفيذ (Running).

- يختلف طول انفجار المعالج باختلاف نوع العملية، العمليات الحسابية (CPU-Intensive) مثلاً : معالجة الصور لها انفجارات معالج طويلة، بينما العمليات التفاعلية (مثل المتصفحات) لها انفجارات قصيرة.
- يتم قياس طول انفجار المعالج بالمللي ثانية.
- معرفة (طول انفجار المعالج) تساعد المجدول على تحديد أولويات العمليات واختيار الخوارزمية المناسبة.

مثال عملي:

- تحتاج عملية لحساب الأعداد الأولية إلى انفجار معالج طويل (مثل 50 مللي ثانية).
- تحتاج عملية تحرير نص (ضغطة مفتاح) إلى انفجار معالج قصير (مثل 5 مللي ثانية).

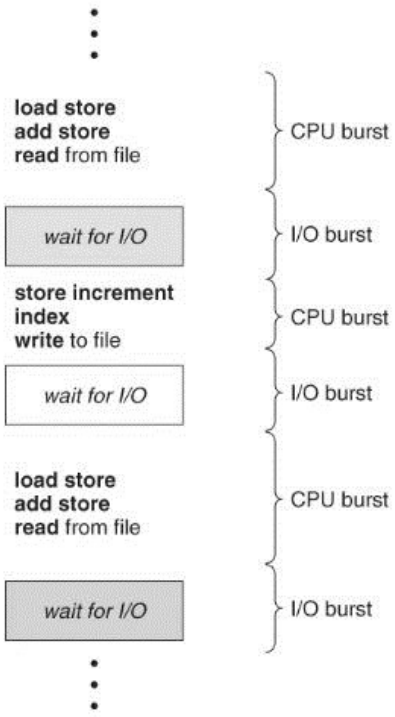
2. انفجار الإدخال / الإخراج (I/O Burst)

هو الفترة الزمنية التي تنتظر فيها العملية اكتمال عملية إدخال/إخراج، مثل (قراءة ملف من القرص أو إرسال بيانات عبر الشبكة)، وتكون العملية خلال هذه الفترة في حالة الانتظار (Waiting).

- يحدث عندما تحتاج العملية إلى التفاعل مع أجهزة الإدخال/الإخراج (مثل القرص الصلب، الطابعة).
- غالباً يكون أطول بكثير من انفجار المعالج بسبب بطء أجهزة الإدخال/الإخراج مقارنة بالمعالج.
- لتجنب إهدار وقت المعالج تتولى جدولة المعالج التبديل بين العمليات خلال انفجارات الإدخال/الإخراج.

مثال عملي:

- تحتاج عملية كتابة البيانات إلى القرص الصلب انفجار إدخال/إخراج يقدر بـ 100 مللي ثانية.



3. المجدول Scheduler

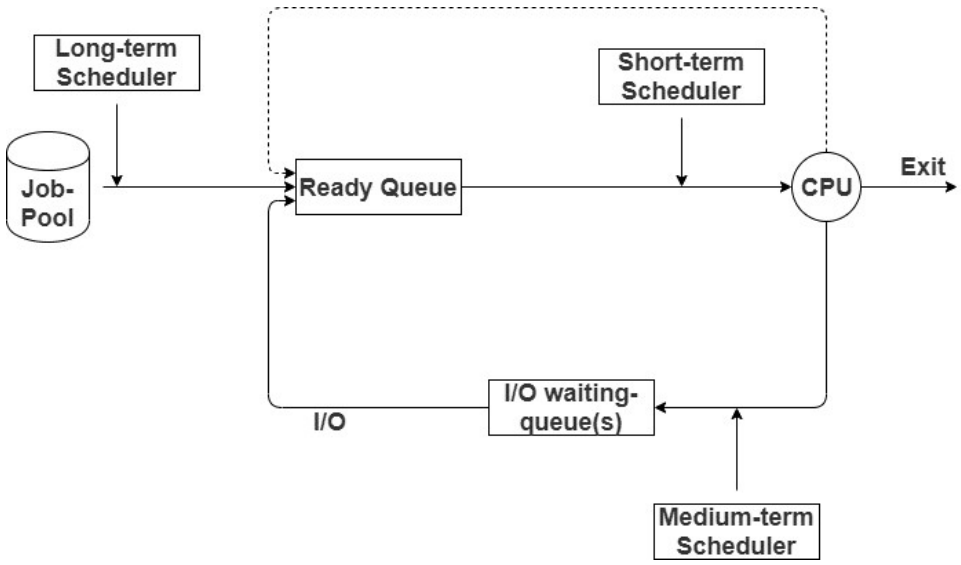
المجدول هو مكون في نواة نظام التشغيل مسؤول عن اختيار العملية التي ستُنفذ على المعالج في أي لحظة زمنية، ويعتمد المجدول على خوارزميات الجدولة لاتخاذ هذه القرارات المناسبة.

وظائف المجدول:

a) تحديد أولويات العمليات بناءً على خوارزميات الجدولة (مثل الأولوية، الوقت المتبقي).
b) إدارة التبديل بين العمليات عند انتهاء انفجار معالج أو حدوث انقطاع (Interrupt).
c) تحقيق أهداف النظام مثل (زيادة الإنتاجية، تقليل وقت الانتظار، أو تحسين الاستجابة).

أنواع المجدولات:

1. المجدول طويل المدى (Long-Term Scheduler): يتحكم في قبول العمليات الجديدة في النظام، ويحدد عدد العمليات التي تدخل طابور الجاهزية (Ready Queue) لتحقيق التوازن بين العمليات الحسابية والتفاعلية.
2. المجدول قصير المدى (Short-Term Scheduler): يختار العملية التالية من طابور الجاهزية لتنفيذها على المعالج، وتقدر سرعة التنفيذ بالميكرو ثانية لتقليل التأخير.
3. المجدول متوسط المدى (Medium-Term Scheduler): يتعامل مع تبديل العمليات (Swapping) بين الذاكرة الرئيسية والثانوية لتحسين الأداء.



مثال عملي:

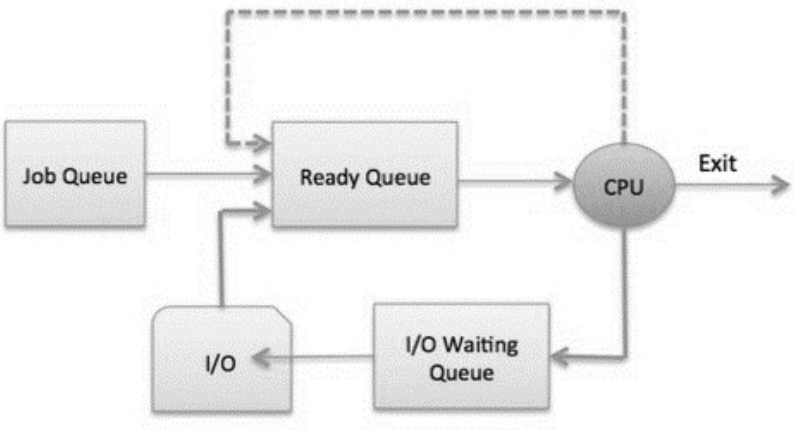
في نظام Linux، المجدول قصير المدى (مثل Completely Fair Scheduler) يختار عملية مثل firefox من طابور الجاهزية لتنفيذها بعد انتهاء انفجار معالج لعملية أخرى مثل gcc.

4. طوابير الجدولة Scheduling Queues

هي هياكل بيانات تُستخدم لتخزين العمليات في حالات مختلفة (جاهزة، قيد التنفيذ، في انتظار) أثناء إدارتها بواسطة المجدول.

أنواع الطوابير:

1. طابور الوظائف (Job Queue): يحتوي على جميع العمليات في النظام، بما في ذلك العمليات الجديدة التي لم تُقبل بعد.
2. طابور الجاهزية (Ready Queue): يحتوي على العمليات الجاهزة للتنفيذ، في انتظار تخصيص المعالج، يتم تنظيمه بناءً على أولويات العمليات أو خوارزمية الجدولة.
3. طابور الإدخال/الإخراج (I/O Queue): يحتوي على العمليات التي تنتظر اكتمال عمليات الإدخال/الإخراج.



آلية العمل:

- عند إنشاء عملية جديدة، تُضاف إلى طابور الوظائف، ثم ينقلها المجدول طويل المدى إلى طابور الجاهزية.
- يختار المجدول قصير المدى عملية من طابور الجاهزية لتنفيذها.
- تُنقل العملية إلى طابور الإدخال/الإخراج عندما تحتاج لإدخال أو إخراج بيانات، ثم تعود إلى طابور الجاهزية بعد ذلك.

مثال عملي:

في نظام Windows، عند تشغيل تطبيقات مثل Chrome وNotepad، يتم وضعها في طابور الجاهزية. إذا قام Chrome بطلب صفحة ويب (I/O Burst)، يُنقل إلى طابور الإدخال/الإخراج، بينما يُخصص المعالج لـ Notepad.

5. الجدولة الاستباقية وغير الاستباقية (Preemptive vs. Non-Preemptive Scheduling)

جدولة المعالج يمكن أن تكون إما استباقية أو غير استباقية بناءً على كيفية التعامل مع تخصيص المعالج والتبديل بين العمليات.

الجدولة غير الاستباقية (Non-Preemptive Scheduling):

عند تخصيص المعالج لعملية ما، فإن هذه العملية تستمر في التنفيذ حتى تنتهي (أي حتى يكتمل انفجار المعالج) أو حتى تدخل العملية طواعية في حالة انتظار، مثل انتظار إدخال / إخراج.

يتميز هذا النوع من الجدولة بأنه:

- لا يتم مقاطعة العملية أثناء تنفيذها، مما يقلل من تكلفة التبديل بين العمليات (Context Switching).
- يُستخدم في الأنظمة البسيطة أو التطبيقات التي تتطلب إكمال المهام دون انقطاع.

من عيوبه أنه:

- لا يناسب الأنظمة التفاعلية، فقد تؤدي عملية طويلة إلى تأخير العمليات الأخرى Convoy Effect.
- أداء ضعيف في الأنظمة متعددة المهام.

أمثلة على خوارزميات الجدولة غير الاستباقية:

- First-Come, First-Served (FCFS): تُنفذ هذه الخوارزمية العمليات حسب ترتيب وصولها.
- Shortest Job First (SJF): تُختار هذه الخوارزمية العملية ذات الانفجار الأقصر للمعالج.

مثال عملي:

في نظام دفعي Batch System، لا يتم تخصيص المعالج لعملية أخرى حتى إنتهاء تنفيذ العملية الحسابية.

الجدولة الاستباقية (Preemptive Scheduling):

في هذا النوع من الجدول يمكن للمجدول مقاطعة العملية قيد التنفيذ وتخصيص المعالج لعملية أخرى بناءً على (أولوية العمليات، انتهاء الشريحة الزمنية المخصصة، أو حدوث انقطاع).

يتميز هذا النوع من الجدولة بأنه:

- يُستخدم في الأنظمة التفاعلية متعددة المهام (مثل Windows، Linux) لضمان استجابة سريعة.
- يحتاج إلى تبديل السياق Context Switching بين العمليات، مما يزيد من التكلفة الحسابية.

من عيوبها:

- تعقيد تنفيذ الجدولة الاستباقية مقارنة بالجدولة غير الاستباقية.
- زيادة التكلفة بسبب التبديل بين العمليات.

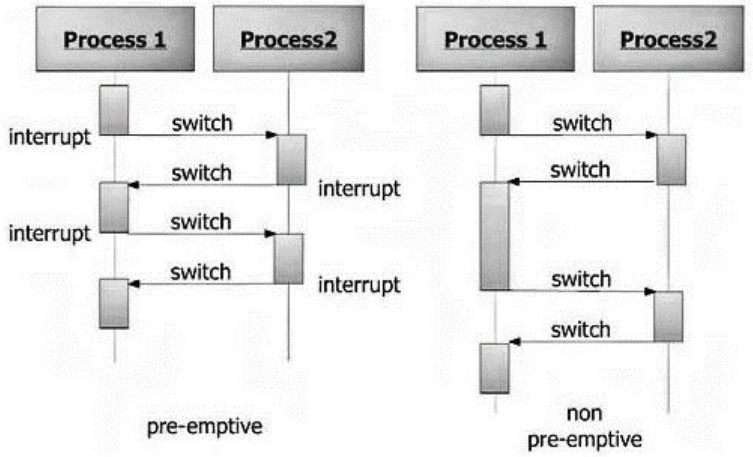
أمثلة على خوارزميات الجدولة الاستباقية:

- Round Robin: تُخصص شريحة زمنية لكل عملية، مع مقاطعة العملية إذا لم تنته خلال الشريحة.
- Priority Scheduling: تُنفذ العملية ذات الأولوية الأعلى، مع إمكانية مقاطعة العمليات ذات الأولوية المنخفضة.

مثال عملي:

في نظام Linux، إذا كانت عملية firefox قيد التنفيذ، ووصلت عملية ذات أولوية أعلى (عملية نظام مثلاً)، يقوم المجدول بمقاطعة firefox وتخصيص المعالج للعملية الجديدة.

مقارنة بين الجدولة الاستباقية وغير الاستباقية



معيار المقارنة	غير استباقية	استباقية
اسلوب العمل	العملية تُنفيذ دون مقاطعة	يمكن مقاطعة العملية
التكلفة الحسابية	منخفضة لا يوجد تبديل	مرتفعة بسبب التبديل بين العمليات
الاستجابة	بطيئة، مع العمليات الطويلة	سريعة وتناسب الأنظمة التفاعلية
الخوارزميات	FCFS ،SJF	Round Robin ،Priority Scheduling
التطبيقات	أنظمة دفعية، تطبيقات بسيطة	أنظمة متعددة المهام، أنظمة تفاعلية
التعقيد	بسيطة	معقدة

مثال عملي عن الجدولة في نظام تشغيل Windows

- يدير نظام Windows عمليات مثل chrome.exe، notepad.exe.
- قد يحتاج chrome لتحديث صفحة ويب (أي يحتاج انفجار معالج قصير 10 مللي ثانية).
- ينتظر chrome تحميل صفحة ويب من الخادم 100 مللي ثانية (انفجار إدخال/ إخراج).
- عندئذ يستخدم المجدول قصير المدى جدولة استباقية لتخصيص المعالج إلى notepad أثناء انتظار chrome للإدخال/الإخراج.
- فينتقل chrome إلى طابور الإدخال/الإخراج ، بينما يصبح notepad في طابور الجاهزية.

5.1. خوارزمية FCFS – First-Come, First-Served :

هي خوارزمية جدولة غير استباقية Non-Preemptive، حيث تُنفذ العمليات حسب ترتيب وصولها إلى طابور الجاهزية، وعند تخصيص المعالج لعملية، فإن العملية تستمر في التنفيذ حتى تكتمل.

آلية العمل:

- يتم وضع العمليات في طابور الجاهزية بناءً على وقت وصولها.
- تُختار العملية الأولى في الطابور للتنفيذ.
- بعد اكتمال العملية، تُختار العملية التالية في الطابور، وهكذا.

المزايا

- البساطة: سهلة الفهم والتطبيق، مما يجعلها مناسبة للأنظمة البسيطة.
- العدالة: تضمن تنفيذ العمليات حسب ترتيب وصولها، مما يحقق عدالة أساسية.
- تكلفة منخفضة: لا تتطلب تبديل سياق (Context Switching) إلا عند اكتمال العملية، مما يقلل التكلفة الحسابية.

العيوب

- تأثير القافلة Convoy Effect إذا وصلت عملية ذات انفجار معالج طويل أولاً، ستؤخر العمليات الأخرى ذات الانفجارات الأقصر، مما يزيد متوسط زمن الانتظار.
- غير مناسبة للأنظمة التفاعلية: لا تدعم الاستجابة السريعة للعمليات التفاعلية (مثل تطبيقات المستخدم).
- أداء ضعيف مع العمليات المتفاوتة: فإذا اختلفت أطوال انفجارات المعالج بشكل كبير، فإن ذلك يمكن أن يؤدي إلى أداء غير مثالي.

مصطلحات أساسية:

1. وقت الوصول Arrival Time اللحظة التي تصل فيها العملية إلى طابور الجاهزية.
2. زمن انفجار المعالج Burst Time الوقت المطلوب لتنفيذ العملية على المعالج.
3. زمن الإكمال Completion Time اللحظة التي تكتمل فيها العملية، أي بعد تنفيذ انفجار المعالج.

4. زمن الانتظار Waiting Time الوقت الذي تقضيه العملية في طابور الجاهزية قبل تخصيص المعالج لها.

يحسب وفق المعادلة التالية: زمن الانتظار = زمن البدء – وقت الوصول.

5. زمن الإنجاز Turnaround Time الوقت الكلي من وصول العملية حتى اكتمالها.

يحسب وفق المعادلة التالية: زمن الإنجاز = زمن الإكمال – وقت الوصول.

6. متوسط زمن الانتظار لجميع العمليات = (مجموع أزمنة الانتظار) ÷ (عدد العمليات)

7. متوسط زمن الإكمال لجميع العمليات = (مجموع أزمنة الإكمال) ÷ (عدد العمليات)

أمثلة عملية

المثال الأول: عمليات بأوقات وصول متساوية

لنفترض أن لدينا ثلاث عمليات P1 ، P2، P3 تصل جميعها في الوقت 0 أي أن وقت الوصول = 0، مع أزمنة انفجار معالج مختلفة موضحة بالجدول التالي:

العملية	وقت الوصول	زمن انفجار المعالج
P1	0	6
P2	0	3
P3	0	1

الخطوة 1: ترتيب العمليات

- بما أن جميع العمليات تصل في الوقت 0، يمكن اختيار أي ترتيب في طابور الجاهزية، لنفترض أن الترتيب P1 → P2 → P3

الخطوة 2: رسم خط زمني (Gantt Chart)

0	6	9	10
P1		P2	P3

- تُنفذ P1 من الوقت 0 إلى 6 (زمن انفجار = 6).
- تُنفذ P2 من الوقت 6 إلى 9 (زمن انفجار = 3).
- تُنفذ P3 من الوقت 9 إلى 10 (زمن انفجار = 1).

الخطوة 3: حساب زمن الإكمال

- زمن الإكمال P1 = 6
- زمن الإكمال P2 = 9
- زمن الإكمال P3 = 10

الخطوة 4: حساب زمن الانتظار

- زمن الانتظار P1 = زمن البدء – وقت الوصول = 0 – 0 = 0
- زمن الانتظار P2 = زمن البدء – وقت الوصول = 6 – 0 = 6
- زمن الانتظار P1 = زمن البدء – وقت الوصول = 9 – 0 = 9

الخطوة 5: حساب المتوسطات

- متوسط زمن الانتظار = (0 + 6 + 9) ÷ 3 = 15 ÷ 3 = 5 وحدات زمنية
- متوسط زمن الإكمال = (6 + 9 + 10) ÷ 3 = 25 ÷ 3 = 8.33 وحدات زمنية

التفسير

- P1 لم تنتظر لأنها نفذت أولاً
- P2 انتظرت 6 وحدات زمنية حتى انتهاء P1
- P3 انتظرت 9 وحدات زمنية حتى انتهاء P1 وP2، وهذا يُظهر تأثير العمليات الطويلة على زمن الانتظار.

المثال الثاني: عمليات بأوقات وصول مختلفة

لنفترض أن لدينا ثلاث عمليات مع أوقات وصول، وأزمنة انفجار معالج، موضحة في الجدول التالي:

العملية	وقت الوصول	زمن انفجار المعالج
P1	0	5
P2	1	3
P3	2	2

الخطوة 1: ترتيب العمليات

- في الوقت 0، P1 فقط متاحة لذا تُنفذ.
- في الوقت 1، P2 تصل لكن P1 لا تزال قيد التنفيذ (غير استباقية).
- في الوقت 2، P3 تصل لكن P1 لا تزال قيد التنفيذ.
- بعد انتهاء P1 ، تُنفذ P2 لأنها وصلت قبل P3 ، ثم تنفذ P3

الخطوة 2: رسم خط زمني

0	5	8	10
P1		P2	P3

- P1 (زمن انفجار = 5) ◄ تُنفذ من الوقت 0 إلى 5.
- P2 (زمن انفجار = 3) ◄ تُنفذ من الوقت 5 إلى 8.
- P3 (زمن انفجار = 2) ◄ تُنفذ من الوقت 8 إلى 10.

الخطوة 3: حساب زمن الإكمال

- زمن الإكمال لـ P1 = 5
- زمن الإكمال لـ P2 = 8
- زمن الإكمال لـ P3 = 10

الخطوة 4: حساب زمن الانتظار

- زمن الانتظار P1 = زمن البدء – وقت الوصول = 0 – 0 = 0
- زمن الانتظار P2 = زمن البدء – وقت الوصول = 5 – 1 = 4
- زمن الانتظار P3 = زمن البدء – وقت الوصول = 8 – 2 = 6

الخطوة 5: حساب المتوسطات

- متوسط زمن الانتظار = (0 + 4 + 6) ÷ 3 = 10 ÷ 3 = 3.33 وحدات زمنية
- متوسط زمن الإكمال = (5 + 8 + 10) ÷ 3 = 23 ÷ 3 = 7.67 وحدات زمنية

التفسير

- P1 نفذت فور وصولها، لذا لم تنتظر.
- P2 انتظرت حتى انتهاء P1 ، مما تسبب في زمن انتظار 4 وحدات.
- P3 انتظرت حتى انتهاء P1 وP2، مما زاد زمن انتظارها إلى 6 وحدات.
- تأثير القافلة واضح هنا، حيث تسببت P1 (ذات انفجار طويل) في تأخير P2 وP3.

تطبيقات FCFS في نظم التشغيل

1. تُستخدم FCFS في الأنظمة الدفعية Batch Systems التي تُنفذ المهام بشكل تسلسلي.
2. إدارة الطوابير في الأجهزة مثل طوابير الطباعة، حيث تُطبع الوثائق حسب ترتيب طلبها.
3. تستخدم في الأجهزة ذات المهام المحدودة (الأنظمة المدمجة) التي لا تتطلب استجابة سريعة.

5.2. خوارزمية الجدولة الدورية (Round Robin)

هي خوارزمية جدولة استباقية Preemptive Scheduling تُخصص شريحة زمنية ثابتة Time Slice لكل عملية في طابور الجاهزية Ready Queue ، يتم تنفيذ العمليات بشكل دوري، حيث تُمنح كل عملية شريحة زمنية واحدة، ثم تُعاد إلى نهاية الطابور إذا لم تكتمل، مما يضمن توزيعاً عادلاً لوقت المعالج، تُستخدم على نطاق واسع في الأنظمة التفاعلية (مثل أنظمة سطح المكتب والخوادم) لأنها توفر استجابة سريعة وتمنع العمليات الطويلة من احتكار المعالج.

الخصائص الأساسية:

- الاستباقية: إذا تجاوزت العملية شريحة الزمن دون اكتمال، تُقاطع وتُعاد إلى طابور الجاهزية.
- الشريحة الزمنية: مدة زمنية محددة (عادةً 10–100 مللي ثانية) يتم تخصيصها لكل عملية.
- العدالة: تضمن أن كل عملية تحصل على فرصة للتنفيذ في وقت معقول.
- الطابور الدوري: يتم تنظيم العمليات في طابور دائري(Circular Queue) ، حيث تُعاد العمليات غير المكتملة إلى نهاية الطابور.

مثال بسيط:

لدينا ثلاث عمليات P1 تتطلب 20 مللي ثانية، P2 تتطلب 10 مللي ثانية، P3 تتطلب 30 مللي ثانية.

الشريحة الزمنية المخصصة = 10 مللي ثانية.

التنفيذ:

- P1 : 10 مللي ثانية
- P2 : 10 مللي ثانية وتنتهي.
- P3 : 10 مللي ثانية.
- P1 : 10 مللي ثانية وتنتهي.
- P3 : 20 مللي ثانية وتنتهي.

آلية عمل الجدولة الدورية

- تنظيم طابور الجاهزية: تُوضع العمليات الجاهزة في طابور دائري.
- تخصيص شريحة زمنية: يختار المجدول العملية الأولى في الطابور ويُخصص لها شريحة زمنية.
- إذا اكتملت العملية خلال الشريحة الزمنية، تُنهى وتُزال من الطابور.
- إذا لم تكتمل، تُقاطع عند انتهاء الشريحة الزمنية وتُعاد إلى نهاية الطابور.

التبديل بين العمليات Context Switching

- يحفظ المجدول حالة العملية الحالية (مثل سجلات المعالج) ويُحمّل حالة العملية التالية.
- يستمر المجدول في اختيار العمليات من بداية الطابور بشكل دوري حتى تنتهي جميع العمليات.

دور شريحة الزمن Time Slice

- شريحة زمنية قصيرة جداً (مثل 1 مللي ثانية) تؤدي إلى تبديل متكرر بين العمليات، مما يزيد تكلفة التبديل .
- شريحة زمنية طويلة جداً (مثل 500 مللي ثانية) تقلل التبديل ولكن قد تؤخر استجابة العمليات التفاعلية.

- الشريحة المثالية تعتمد على طبيعة النظام، وعادةً تتراوح بين 10–100 مللي ثانية لتحقيق توازن بين الاستجابة والكفاءة.

المزايا

- العدالة: كل عملية تحصل على شريحة زمنية متساوية، مما يمنع احتكار المعالج.
- الاستجابة السريعة: مناسبة للأنظمة التفاعلية (مثل أنظمة سطح المكتب) لأن العمليات تحصل على وقت معالج بسرعة.
- البساطة: سهلة التنفيذ مقارنة بخوارزميات معقدة مثل Shortest Remaining Time First
- مناسبة للأنظمة متعددة المهام: تدعم تشغيل العديد من العمليات في وقت واحد دون تأخير كبير.
- المرونة: يمكن تعديل شريحة الزمن لتتناسب مع احتياجات النظام.

العيوب

- تكلفة التبديل Context Switching Overhead : إذا كانت شريحة الزمن قصيرة جداً، يزداد عدد التبديلات، مما يُهدر وقت المعالج.
- أداء ضعيف مع العمليات الطويلة: العمليات ذات انفجار معالج طويل (CPU Burst) قد تتطلب عدة دورات، مما يزيد زمن الإنجاز.
- حساسية شريحة الزمن: اختيار شريحة زمنية غير مناسبة (قصيرة أو طويلة) يؤثر سلباً على الأداء.
- عدم مراعاة الأولويات: لا تأخذ في الاعتبار أولويات العمليات، مما قد يؤخر العمليات الحرجة.
- زمن انتظار مرتفع في بعض الحالات: إذا كان هناك العديد من العمليات، قد تنتظر العمليات وقتاً طويلاً في الطابور.

التطبيقات

- أنظمة تفاعلية، متعددة المهام
- أنظمة دفعية

مثال عملي

العمليات: P1 20 مللي ثانية، P2 10 مللي ثانية، P3 30 مللي ثانية.

شريحة الزمن: 10 مللي ثانية.

جدول التنفيذ:

- P1 : 0–10 ، المتبقي 10
- P2 : 10–20 تنتهي
- P3 : 20–30 ، المتبقي 20
- P1 : 30–40، تنتهي
- P3 : 40–50، المتبقي 10
- P3 :50–60، تنتهي

زمن الانتظار:

- P1: (20–10) + (40–30) = 20
- P2: 10
- P3: (30–20) + (50–40) = 20

متوسط زمن الانتظار: (20 + 10 + 20) / 3 = 16.67 مللي ثانية.

زمن الإنجاز:

- P1: 40
- P2: 20
- P3: 60

متوسط زمن الإنجاز: (40 + 20 + 60) / 3 = 40 مللي ثانية.

تطبيق عملي

- يدير نظام Windows عمليات مثل chrome، notepad ، cmd
- يستخدم جدولة استباقية تعتمد على الأولويات مع شرائح زمنية ، يشبه Round Robin للعمليات ذات الأولوية المتساوية.
- chrome يحتاج إلى 20 مللي ثانية لتحميل صفحة.
- notepad يحتاج إلى 10 مللي ثانية لمعالجة إدخال نص.
- cmd يحتاج إلى 15 مللي ثانية لتنفيذ أمر.
- شريحة الزمن: 10 مللي ثانية.

chrome (10 مللي ثانية) ← notepad (10 مللي ثانية، تنتهي) ← cmd (10 مللي ثانية) ←
chrome (10 مللي ثانية ، وتنتهي) ← cmd (5 مللي ثانية ، وتنتهي).